10051535.011802

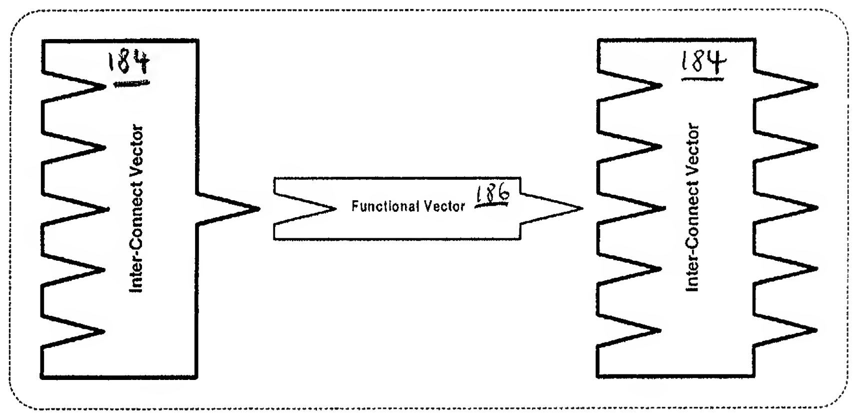


Figure 1A

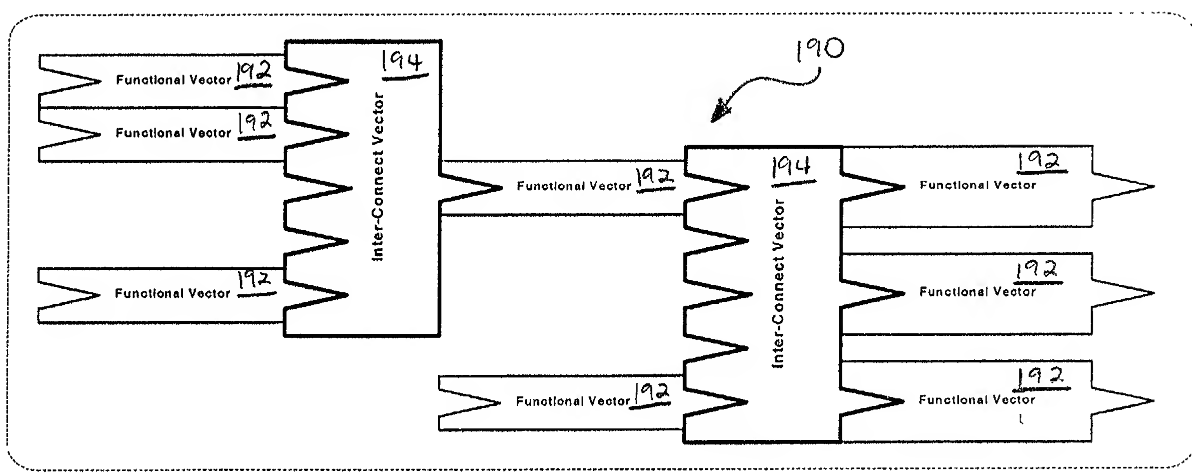


Figure 1B

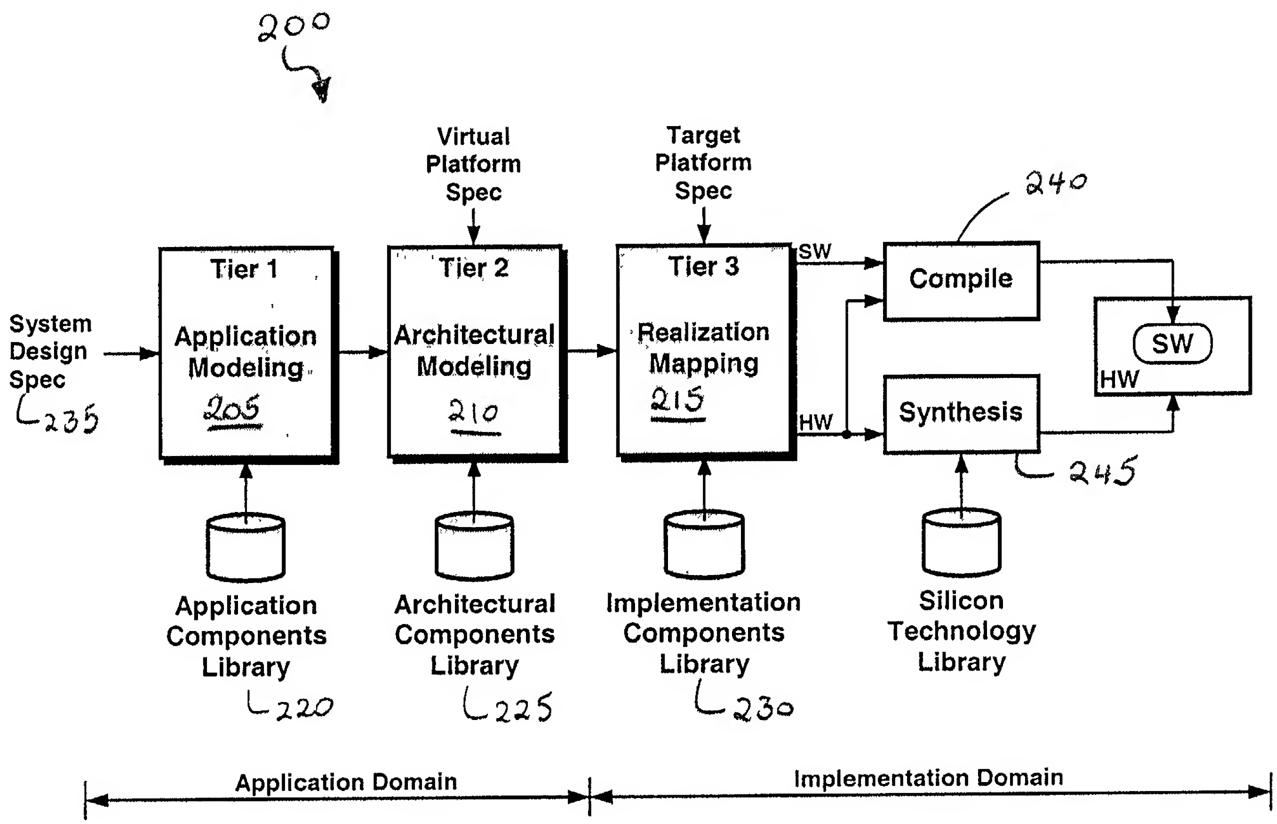

200
System Design Spec
235
Tier 1
Application Modeling
205
Virtual Platform Spec
Tier 2
Architectural Modeling
210
Target Platform Spec
Tier 3
Realization Mapping
215
SW
HW
Compile
240
Synthesis
245
SW
HW
Application Components Library
220
Architectural Components Library
225
Implementation Components Library
230
Silicon Technology Library
Application Domain
Implementation Domain

Figure 2A

20BTTO"SESTSOOT

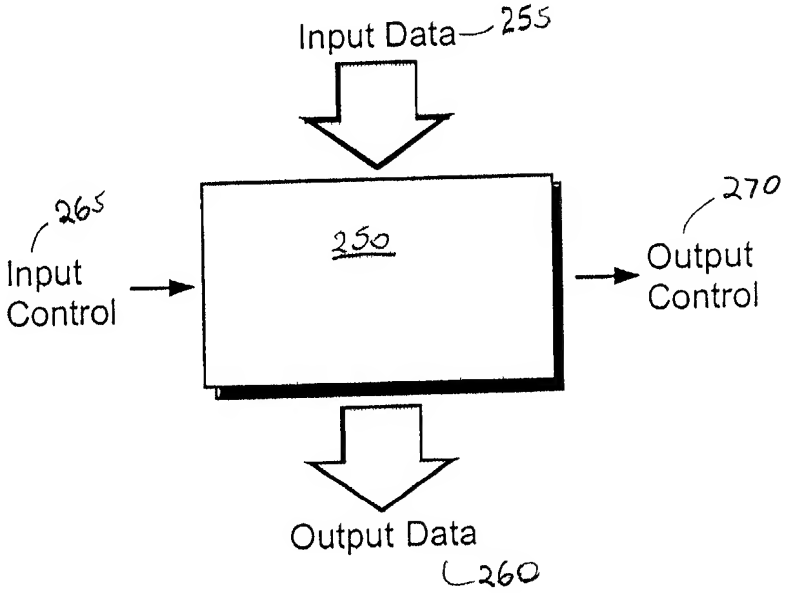


Figure 2B

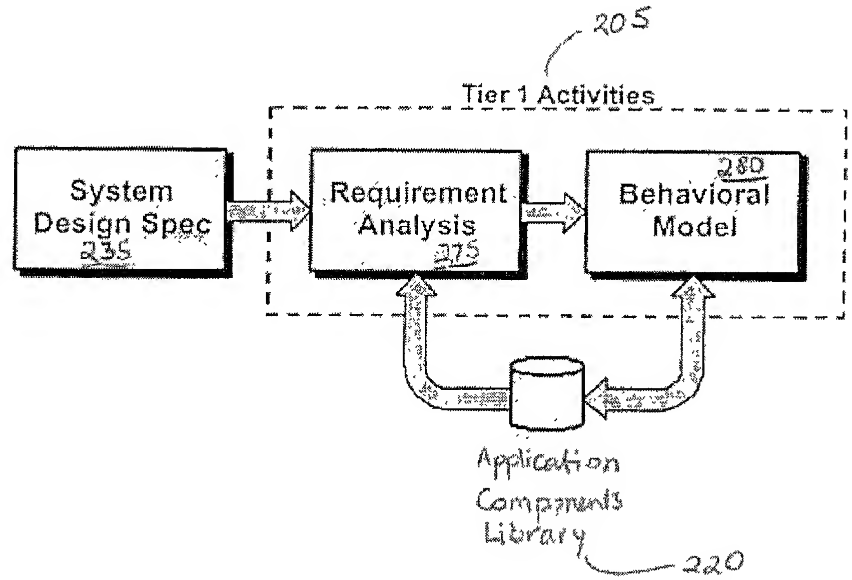

205
Tier 1 Activities
System Design Spec 235
Requirement Analysis 275
Behavioral Model 280
Application Components Library
220

Figure 2C

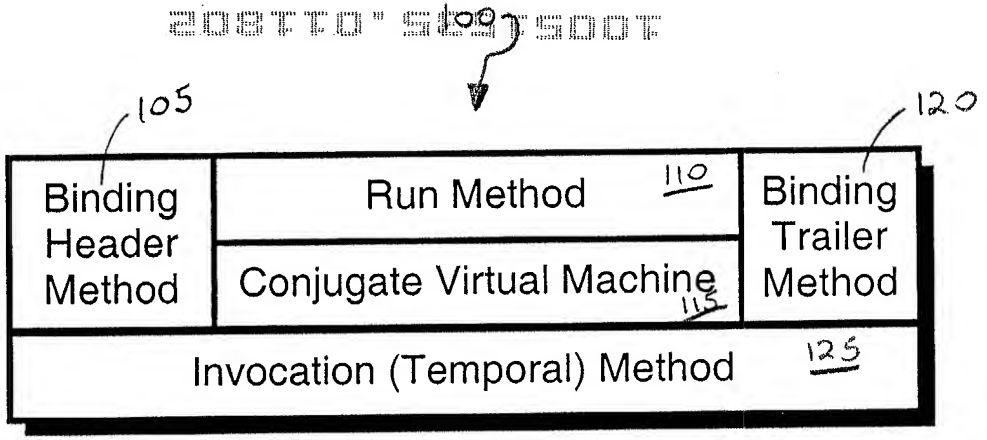

100
105
120
Binding Header Method
Run Method
110
Conjugate Virtual Machine
115
Binding Trailer Method
Invocation (Temporal) Method
125

Figure 3A

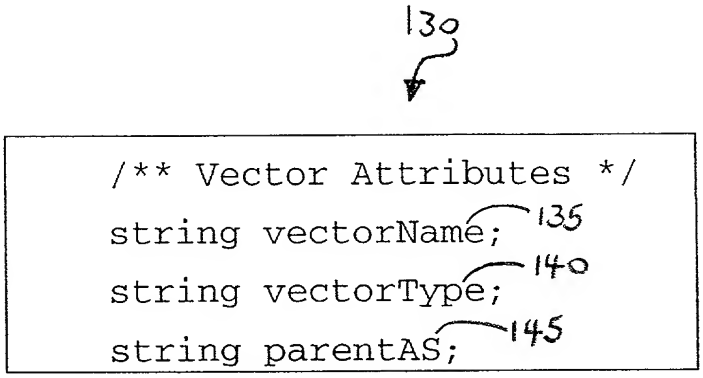


Figure 3B

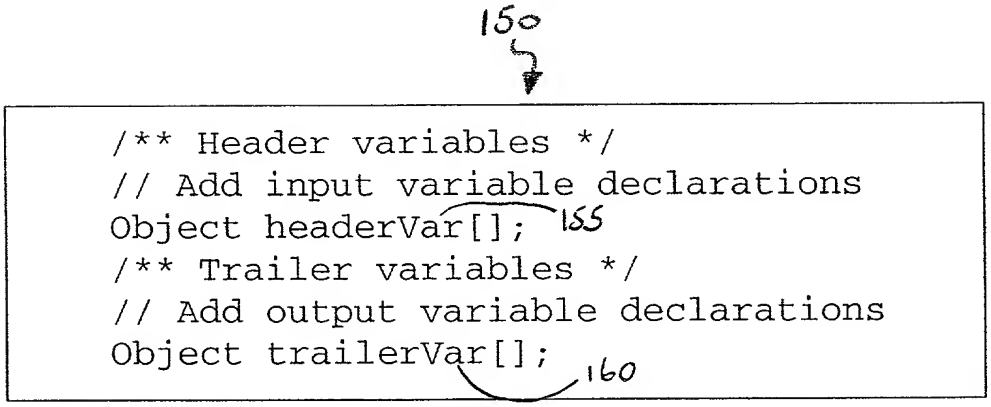
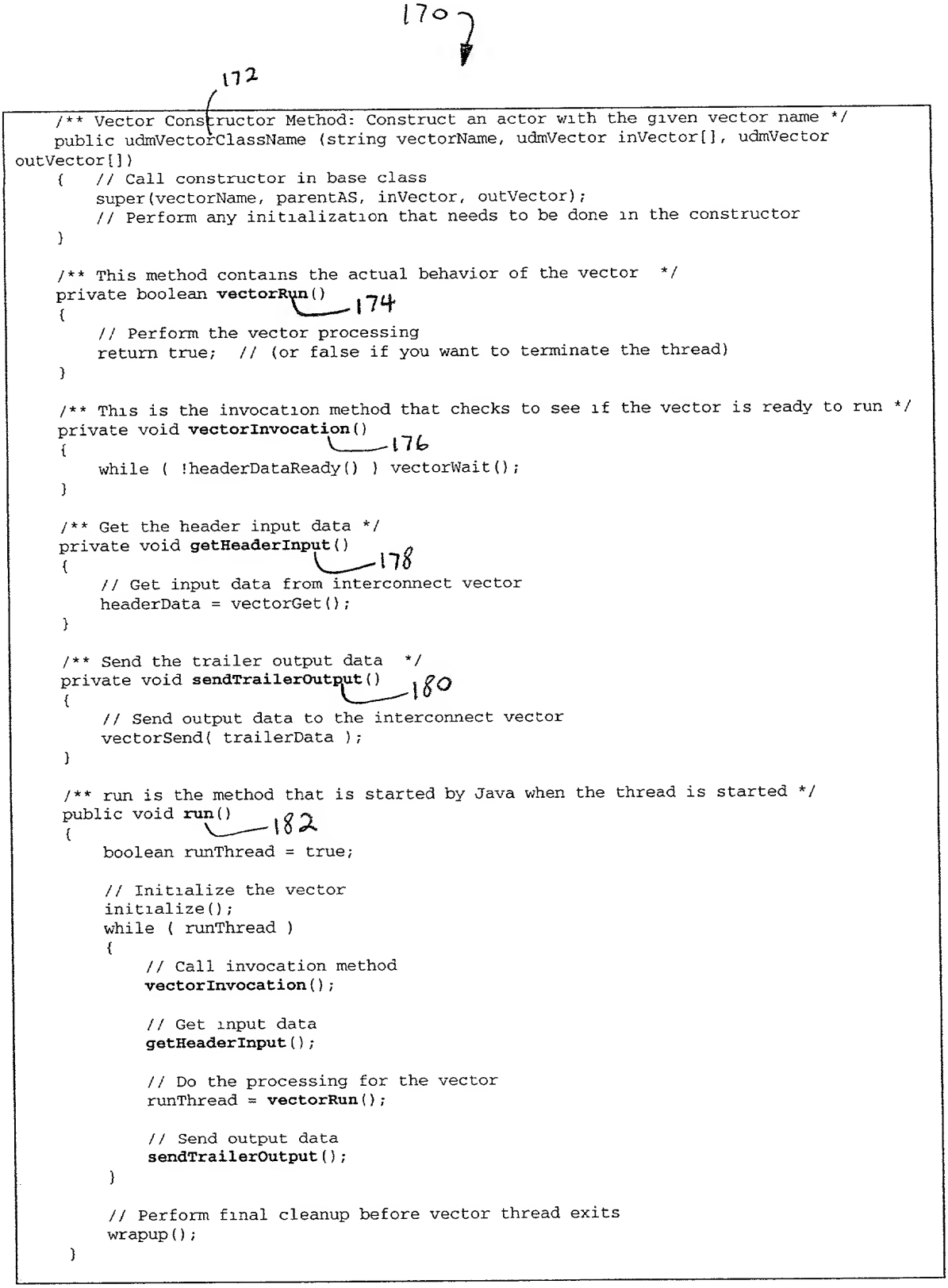


Figure 3C




```
    /** Vector Constructor Method: Construct an actor with the given vector name */
    public udmVectorClassName (string vectorName, udmVector inVector[], udmVector
outVector[])
    {   // Call constructor in base class
        super(vectorName, parentAS, inVector, outVector);
        // Perform any initialization that needs to be done in the constructor
    }

    /** This method contains the actual behavior of the vector  */
    private boolean vectorRun()
    {
        // Perform the vector processing
        return true;  // (or false if you want to terminate the thread)
    }

    /** This is the invocation method that checks to see if the vector is ready to run */
    private void vectorInvocation()
    {
        while ( !headerDataReady() ) vectorWait();
    }

    /** Get the header input data */
    private void getHeaderInput()
    {
        // Get input data from interconnect vector
        headerData = vectorGet();
    }

    /** Send the trailer output data  */
    private void sendTrailerOutput()
    {
        // Send output data to the interconnect vector
        vectorSend( trailerData );
    }

    /** run is the method that is started by Java when the thread is started */
    public void run()
    {
        boolean runThread = true;

        // Initialize the vector
        initialize();
        while ( runThread )
        {
            // Call invocation method
            vectorInvocation();

            // Get input data
            getHeaderInput();

            // Do the processing for the vector
            runThread = vectorRun();

            // Send output data
            sendTrailerOutput();
        }

        // Perform final cleanup before vector thread exits
        wrapup();
    }
```

Figure 3D

20BTTO"SESTSOOT

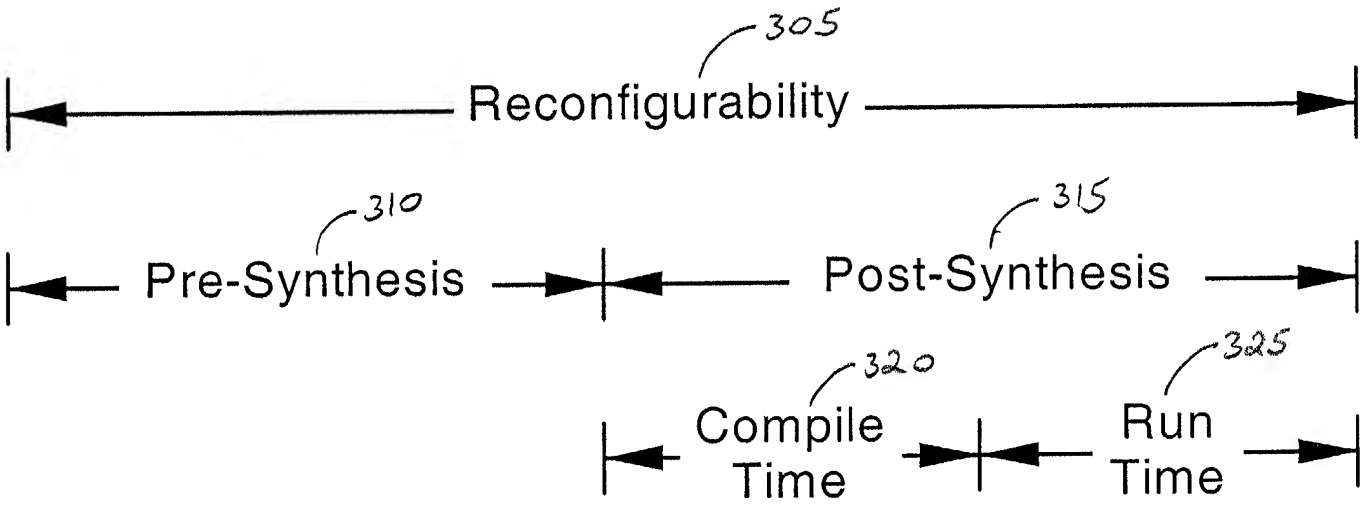


Figure 4